CPE — Como depurar código

Departamento de Engenharia Elétrica – UnB

Simulação de código

- Às vezes (ou muitas vezes?), acontece de programarmos um código e ele não faz o que esperávamos que fizesse.
- Isso acontece por vários motivos, entre os quais destacam-se:
  - Erros de programação: instruções escritas erradas.
  - Erros da nossa lógica: o conjunto de passos pensados que parecia resolver o problema na realidade não cobre todas as situações.
- Eventualmente, simplesmente olhar o código pode não trazer à tona o erro.
- Por isso, utiliza-se uma técnica de simulação do código
  - Pode ser automatizada, utilizando um depurador (*debugger*).
  - Pode ser feita manualmente, utilizando papel e caneta.

Simulação Manual

- Bem simples: Existem apenas 2 passos.
  - "Alocação" dos espaços de variáveis:
  - "Execução" de uma instrução de cada vez: atualizar valores na tabela.
- Alocação de memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
- Após "executar" a linha 1

Tipo	`int`	`int`
Nome	`divisor`	`dividendo`
Valor	?	?

Simulação Manual

- Bem simples: Existem apenas 2 passos.
  - "Alocação" dos espaços de variáveis: fazer uma tabela.
  - "Execução" de uma instrução de cada vez: atualizar valores na tabela.
- Alocação de memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
- Após "executar" a linha 2

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	?	?	?

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;` ← Último executado
    3. `divisor=10;` ← Próximo Comando
    4. `dividendo=13;`
    5. `resultado = dividendo / divisor;`
- Após "executar" a linha 2

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	?	?	?

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
    3. `divisor=10;` ← Último executado
    4. `dividendo=13;` ← Próximo Comando
    5. `resultado = dividendo / divisor;`
- Após "executar" a linha 3

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	~~?~~ 10	?	?

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
    3. `divisor=10;`
    4. `dividendo=13;` ← Último executado
    5. `resultado = dividendo / divisor;` ← Próximo Comando
- Após "executar" a linha 4

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	~~?~~ 10	~~?~~ 13	?

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
    3. `divisor=10;`
    4. `dividendo=13;`
    5. `resultado = dividendo / divisor;` ← Último executado
- Após "executar" a linha 5

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	~~?~~ 10	~~?~~ 13	~~?~~ 1.0

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código:
    1. `int divisor,dividendo;`
    2. `float resultado;`
    3. `divisor=10;`
    4. `dividendo=13;`
    5. `resultado = dividendo / divisor;` ← Último executado
- Término da execução (não há mais comandos)

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	~~?~~ 10	~~?~~ 13	~~?~~ 1.0 ← Erro!

Simulação Manual

- Execução em memória:
  - Ex. Suponha o código (corrigido):
    1. `int divisor,dividendo;`
    2. `float resultado;`
    3. `divisor=10;`
    4. `dividendo=13;`
    5. `resultado = (float)dividendo / (float)divisor;`
- Execução completa

Tipo	`int`	`int`	`float`
Nome	`divisor`	`dividendo`	`resultado`
Valor	~~?~~	~~?~~	~~?~~
	10	13	1.3

Simulação automática

- Se seu programa tem algum erro de execução e você não sabe onde ele está dentro do código-fonte, é possível utilizar o depurador **gdb** que está embutido no Dev-C^{++} para auxiliá-lo.
  1. Primeiro escolha uma linha do seu código onde você deseja que o depurador pare a execução ("breakpoint"), o que é feito clicando na barra preta ao lado da linha escolhida.
  2. Acesse o menu "Debug" e então clique na opção "Debug" (atalho F8). O depurador iniciará a execução do seu programa, e indicará na linha do breakpoint que está parado nela, aguardando seu comando.
  3. Para então começar a executar passo-a-passo seu programa, basta ir apertando o botão "Next Step" ou utilizando o atalho "F7".

Simulação automática

- O depurador possui outras funções como adicionar um “Watch”, ou um vigia para o valor de uma variável ao longo da execução, e também operações para continuar a execução até o próximo breakpoint.
- Para interromper a depuração, basta clicar em “Stop Execution” dentro do menu “Debug”, ou usar o atalho “Ctrl + Alt + F2”.